STRUTS 2

UNIT –IV

ADVANCE JAVA

TOPICS TO BE COVERED

- Basics Of Struts2 : What And Why?
- Model1 V/S Model2
- Struts2 Feature
- Creating Struts2 Application
- Understanding Action Class
- Understanding Struts Xmlfile
- Struts2 Architecture,
- Struts2 Action
- Configuration and Validations.

STRUTS INTRODUCTION

- The **struts 2 framework** is used to develop **MVC-based web application**.
- The struts framework was initially created by **Craig McClanahan** and donated to Apache Foundation in May, 2000 and Struts 1.0 was released in June 2001.
- The current stable release of Struts is Struts 2.3.16.1 in March 2, 2014.

WHAT IS APACHE STRUTS?

- Struts is an open-source framework for building more flexible, maintainable and structured front-ends in Java web applications
- There are two key components in a web application:
  - **the data and business logic performed on this data**
  - **the presentation of data**
- Struts
  - **helps structuring these components in a Java web app.**
  - **controls the flow of the web application, strictly separating these components**
  - **unifies the interaction between them**

- This separation between presentation, business logic and control is achieved by implementing the Model-View-Controller (MVC) Design Pattern

TYPES OF FRAMEWORKS

- Frameworks are divided into 2 types,

Invasive

Non-Invasive

- Invasive means,

➢ it will force the programmers to create their classes by

➢ extending or implementing from per-defined classes or

➢ interfaces provided by that frame work.

- Non-Invasive means

➢ it wont forces the programmer to extend or implement its

➢ own classes or interfaces.

➢ Struts is the type of Invasive frame work.

WHAT IS A WEB FRAMEWORK?

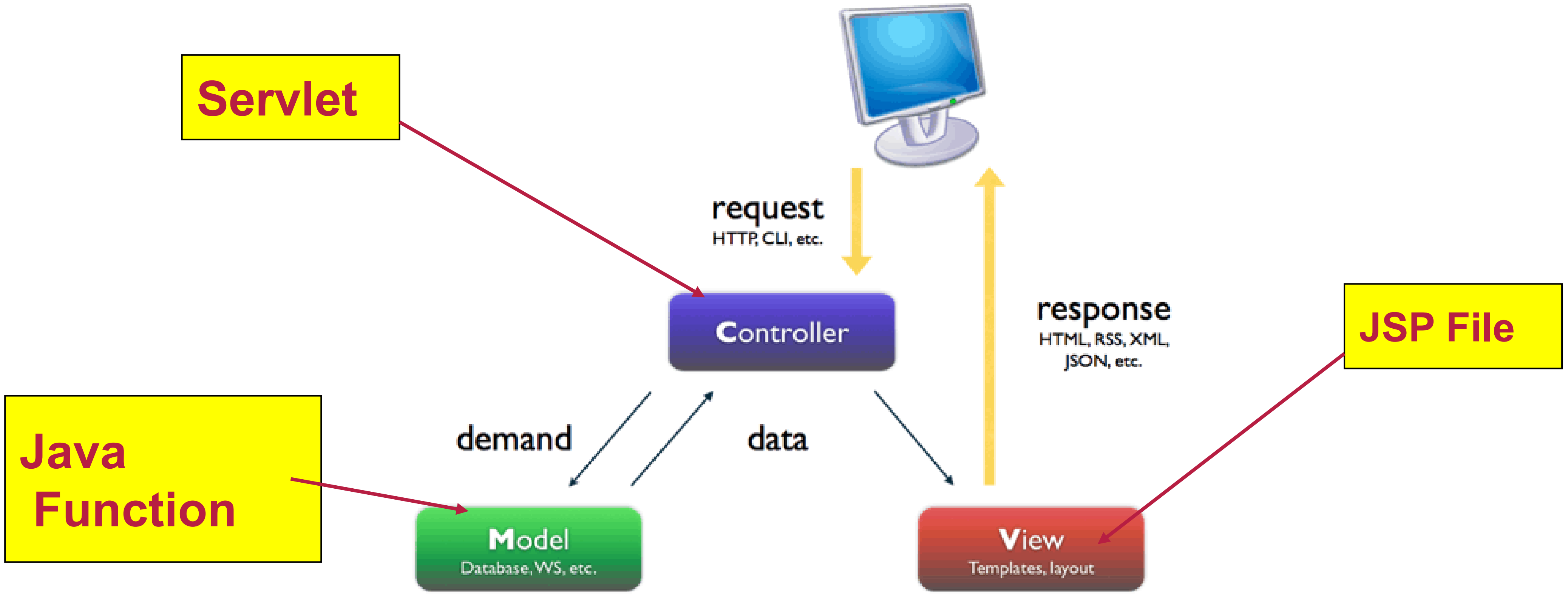
- Web framework is a basic readymade underlying structure, where you have to just add components related to your business.

  - For example, if you take struts, it comes with all jars you might need to develop basic request response cycle, and with basic configuration.
  - It provides the controller servlet or filter which will read your request and convert it to integer or float etc. according to your business requirements.
  - If there is any exception while converting, you don't have to deal with that. Framework deals with the problem and displays you exact message.
  - After all conversion, the framework automatically populate all your data needed from form to the java object.
  - You don't have to write much code to validate all your form data. Frame work provides some basic automatic validations.
  - After you write business logic, you don't have to write code to dispatch request to another page.

WHAT IS A WEB FRAMEWORK?

- It forces the team to implement their code in a standard way. (helps debugging, fewer bugs etc).
  - For Example, in struts immediate backend logic should be in action classes's method. Action class functions intern can call other components to finish business logic.
- Framework might also help you develop complex User Interface easily like iterating tables, the menu, etc (provide some tag libraries)
- Using frameworks like struts 2.0, one need not have to have deep knowledge of Http protocol and its request and responses interfaces to write business logic.

WHY STRUTS?


Servlet
request
HTTP, CLI, etc.
Controller
response
HTML, RSS, XML, JSON, etc.
JSP File
Java Function
demand
data
Model
Database, WS, etc.
View
Templates, layout

WHY STRUTS?

- Traditionally, there are 3 ways to generate dynamic output (typically HTML or XML) in Java web applications:
  - **Servlets**
    - **Java classes with some special methods (**doGet(), doPost(), …**)**
    - **Example:** out.println("<H1>" + myString + "</H1>");
    - **no separation between code and presentation!**
  - **JSPs (Java Server Pages)**
    - **HTML (or other) code with embedded Java code (Scriptlets)**
    - **compiled to Servlets when used for the first time**
    - **Example:** <H1><% out.println(myString); %></H1>
    - **better, but still no separation between code and presentation!**
  - **JSPs with JSTL (JSP Standard Tag Library)**
    - **JSTL defines a set of tags that can be used within the JSPs**
    - **There are tags for iterations, using JavaBeans, printing expressions…**
    - **Example:** <H1><c:out value="${myBean.myString}"/></H1>
    - **better readable and thus better maintainability**

MODEL 1 AND MODEL 2 (MVC) ARCHITECTURE

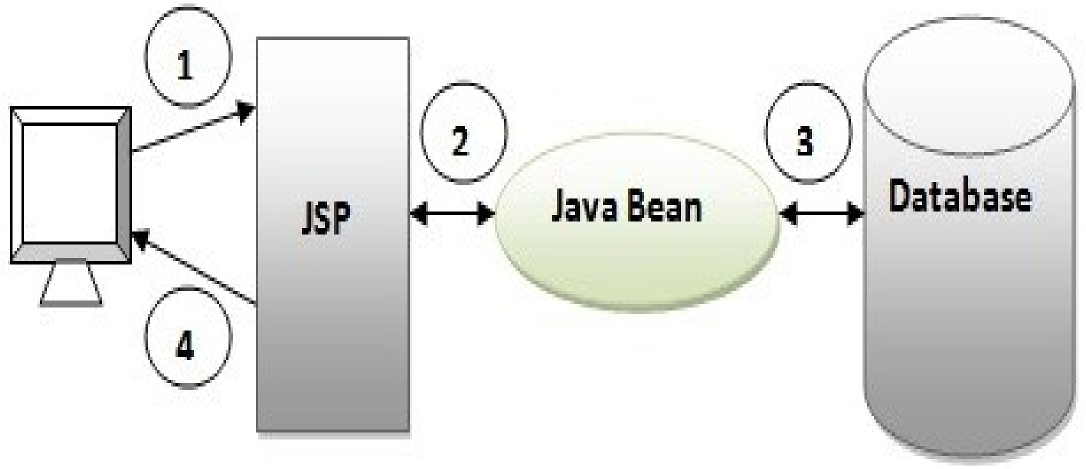
- There are two types of programming models (design models)

1. Model 1 Architecture
2. Model 2 (MVC) Architecture

MODEL 1 ARCHITECTURE

- Servlet and JSP are the main technologies to develop the web applications.
- **Servlet** was considered superior to CGI. Servlet technology doesn't create process, rather it creates thread to handle request. The advantage of creating thread over process is that it doesn't allocate separate memory area. Thus many subsequent requests can be easily handled by servlet.
- **Problem in Servlet technology** Servlet needs to recompile if any designing code is modified. It doesn't provide separation of concern. Presentation and Business logic are mixed up.
- **JSP** overcomes almost all the problems of Servlet. It provides better separation of concern, now presentation and business logic can be easily separated. You don't need to redeploy the application if JSP page is modified. JSP provides support to develop web application using JavaBean, custom tags and JSTL so that we can put the business logic separate from our JSP that will be easier to test and debug.

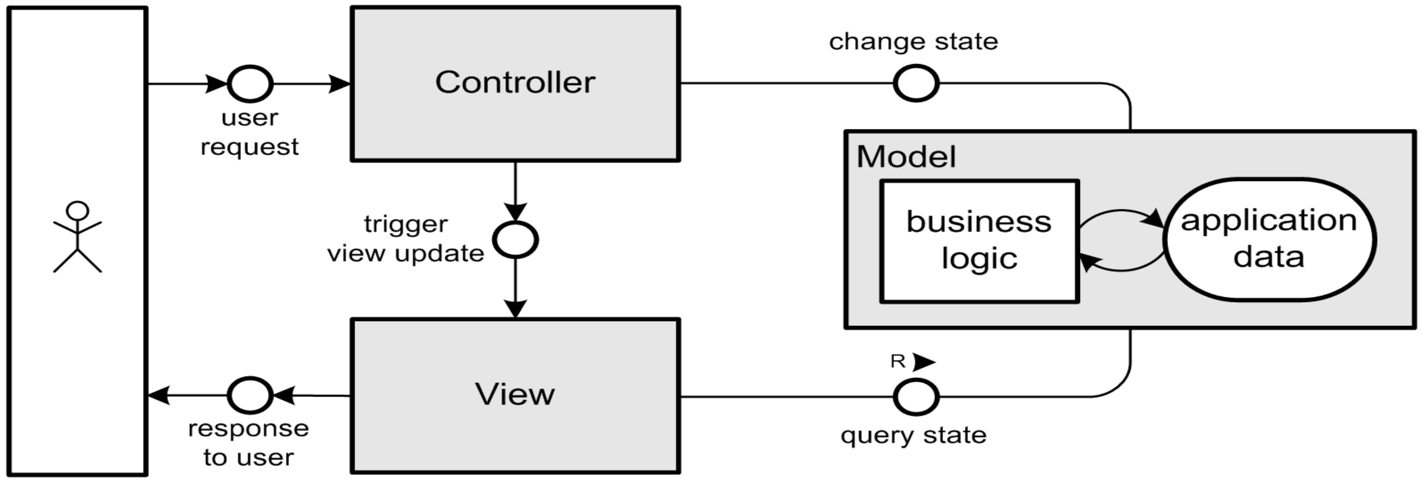
MODEL 1 ARCHITECTURE



1. Browser sends request for the JSP page
2. JSP accesses Java Bean and invokes business logic
3. Java Bean connects to the database and get/save data
4. Response is sent to the browser which is generated by JSP

MODEL 2 (MVC) ARCHITECTURE

- Splits up responsibilities for handling user interactions in an application into three layers:
  - **Model, View, Controller**



- Model
  - **holds application data and business logic**
  - **is absolutely independent from the UIs**

MODEL 2 (MVC) ARCHITECTURE

- View
  - presentation of parts of the Model to the user
  - independent from the internal implementation of the Model
  - there can be different Views presenting the same Model data
- Controller
  - “bridge” between Model and View
  - controls the flow of the application
    - receives/interprets user input
    - performs operations on the Model
    - triggers View update

BENEFITS OF MVC 2

- **Navigation control is centralized** Now only controller contains the logic to determine the next page.
- **Easy to maintain and testability of applications**
- **Easy to extend-**ability to easily develop different kinds of UIs (e.g. console, GUI, …)
- **Better separation of concerns-**separation of different tasks in development
- **Code reusability**

STRUTS 2 FEATURES

1. **Configurable MVC components-** In struts 2 framework, we provide all the components (view components and action) information in struts.xml file. If we need to change any information, we can simply change it in the xml file.
2. **POJO based actions-** In struts 2, action class is POJO (Plain Old Java Object) i.e. a simple java class. Here, you are not forced to implement any interface or inherit any class.
3. **AJAX support**-Struts 2 provides support to ajax technology. It is used to make asynchronous request i.e. it doesn't block the user. It sends only required field data to the server side not all. So it makes the performance fast.

STRUTS 2 FEATURES

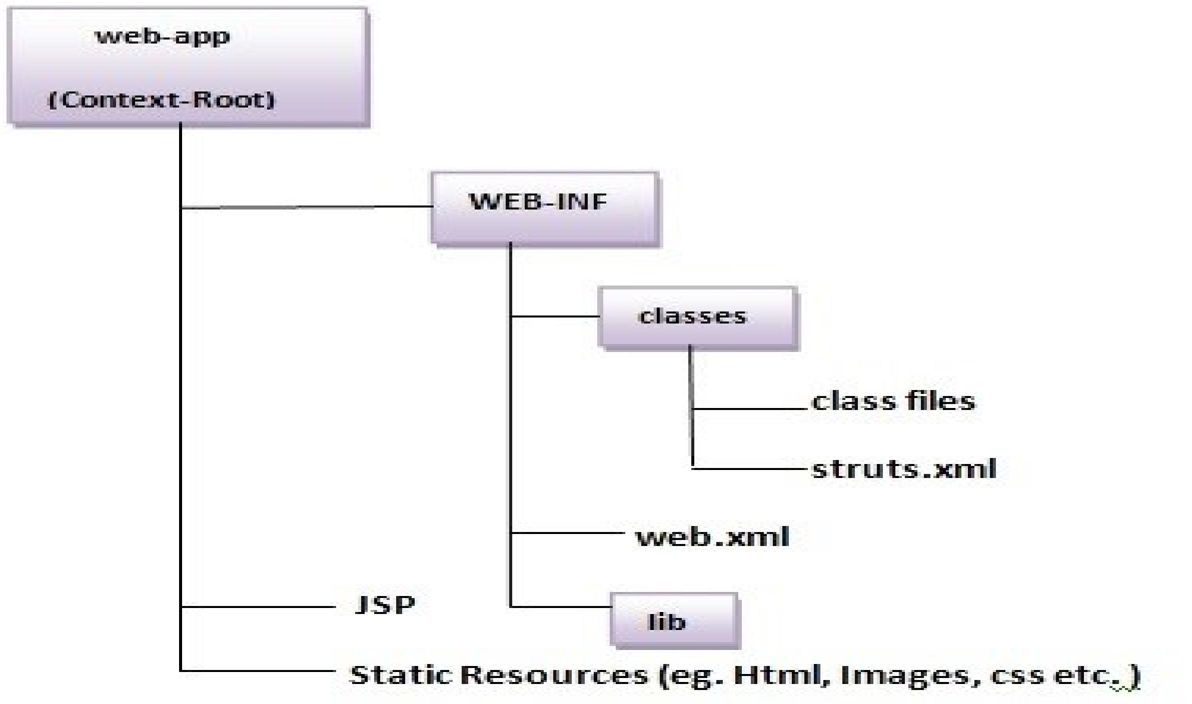
- **Integration Support** -We can simply integrate the struts 2 application with hibernate, spring, tiles etc. frameworks.
- **Various Result Types** - We can use JSP, freemarker, velocity etc. technologies as the result in struts 2.
- **Various Tag support** - Struts 2 provides various types of tags such as UI tags, Data tags, control tags etc to ease the development of struts 2 application.
- **Theme and Template support** –Struts 2 provides three types of theme support: xhtml, simple and css_xhtml. The xhtml is default theme of struts 2. Themes and templates can be used for common look and feel.

CREATING STRUTS 2 APPLICATION (WITHOUT IDE)

1. Create the directory structure
2. Create input page (index.jsp)
3. Provide the entry of Controller in (web.xml) file
4. Create the action class (Product.java)
5. Map the request with the action in (struts.xml) file and define the view components
6. Create view components (welcome.jsp)
7. load the jar files
8. start server and deploy the project

CREATE THE DIRECTORY STRUCTURE

- The directory structure of struts 2 is same as servlet/JSP. Here, struts.xml file must be located in the classes folder.



CREATE INPUT PAGE (INDEX.JSP)

- This jsp page creates a form using struts UI tags. To use the struts UI tags, you need to specify uri /struts-tags. Here, we have used s:form to create a form, s:textfield to create a text field, s:submit to create a submit button.

```
<%@ taglib uri="/struts-tags" prefix="s" %>

<s:form action="product">

<s:textfield name="id" label="Product Id"></s:textfield>

<s:textfield name="name" label="Product Name"></s:textfield>

<s:textfield name="price" label="Product Price"></s:textfield>

<s:submit value="save"></s:submit>

</s:form>
```

PROVIDE THE ENTRY OF CONTROLLER IN (WEB.XML) FILE

- In struts 2, **StrutsPrepareAndExecuteFilter** class works as the controller.
- Struts 2 uses filter for the controller. It is implicitly provided by the struts framework.

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app>
  <filter>
  <filter-name>struts2</filter-name>
   <filter-class>  org.apache.struts2.dispatcher.ng.filter.StrutsPrepareAndExecuteFilter   </filter-class>
  </filter>
  <filter-mapping>
   <filter-name>struts2</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>
</web-app>
```

CREATE THE ACTION CLASS (PRODUCT.JAVA)

- This is simple bean class. In struts 2, action is POJO (Plain Old Java Object). It has one extra method **execute** i.e. invoked by struts framework by default.

```
package com.Welcome;

public class Product {
private int id;
private String name;
private float price;
public int getId() {
    return id;
}
public void setId(int id) {
    this.id = id;
}
public String getName() {
    return name;  }
public void setName(String name) {
    this.name = name;
}
public float getPrice() {
    return price;
}
public void setPrice(float price) {
    this.price = price;
}
  public String execute(){
    return "success";
}
}
```

MAP THE REQUEST IN (STRUTS.XML) FILE AND DEFINE THE VIEW COMPONENTS

- It is the important file from where struts framework gets information about the action and decides which result to be invoked.
- **Struts** element is the root elements of this file. It represents an application.
- **package** element is the sub element of struts. It represents a module of the application. It generally extends the **struts-default** package where many interceptors and result types are defined.
- **action** element is the sub element of package. It represents an action to be invoked for the incoming request. It has name, class and method attributes. If you don't specify name attribute by default execute() method will be invoked for the specified action class.

MAP THE REQUEST IN (STRUTS.XML) FILE AND DEFINE THE VIEW COMPONENTS

- **result** element is the sub element of action. It represents an view (result) that will be invoked. Struts framework checks the string returned by the action class, if it returns success, result page for the action is invoked whose name is success or has no name. It has **name** and **type** attributes. Both are optional. If you don't specify the result name, by default success is assumed as the result name. If you don't specify the type attribute, by default **dispatcher** is considered as the default result type.

MAP THE REQUEST IN (STRUTS.XML) FILE AND DEFINE THE VIEW COMPONENTS

```
 <?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC "-//Apache Software Foundation//DTD Struts
Configuration 2.1//EN" "http://struts.apache.org/dtds/struts-2.1.dtd">
<struts>
<package name="default" extends="struts-default">
  <action name="product" class="com.Welcome.Product">
<result name="success">welcome.jsp</result>
</action>
  </package>
</struts>
```

Struts.xml

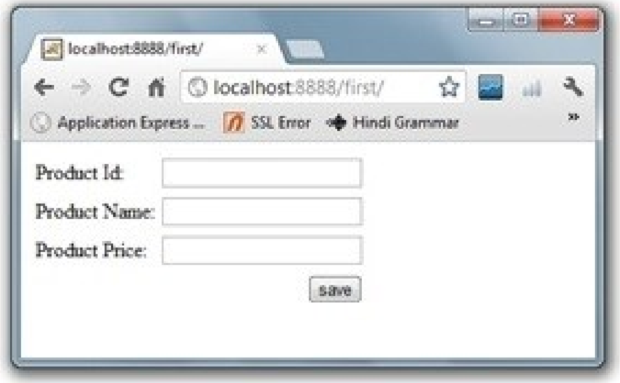
CREATE VIEW COMPONENTS (WELCOME.JSP)

- It is the view component the displays information of the action. Here, we are using struts tags to get the information.
- The s:property tag returns the value for the given name, stored in the action object.

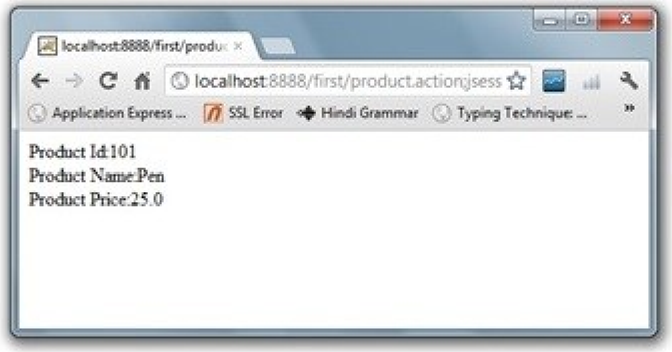
```
<%@ taglib uri="/struts-tags" prefix="s" %>

Product Id:<s:property value="id"/><br/>
Product Name:<s:property
value="name"/><br/>
Product Price:<s:property value="price"/><br/>
```

LOAD THE JAR FILES-START SERVER AND DEPLOY THE PROJECT

- Run this application, you need to have the struts 2 jar files.
- Finally, start the server and deploy the project and access it.



Product Id:
Product Name:
Product Price:
save


localhost:8888/first/produc
localhost:8888/first/product.action;jsess
Application Express ...
SSL Error
Hindi Grammar
Typing Technique: ...

```
Product Id:101
Product Name:Pen
Product Price:25.0
```

CREATING STRUTS2 IN NETBEANS IDE

- Struts application is created in the same way as you create any other web application in the IDE - using the New Web Application wizard, with the additional step of indicating that you want the Struts libraries and configuration files to be included in your application.
- Choose File > New Project (Ctrl-Shift-N) from the main menu → Select Java Web in the list of Categories →select Web Application in the list of Projects→ Click Next.
- In the Name and Location panel, enter MyStrutsApp for Project Name and click Next.
- In the Server and Settings panel, select the server to which you want to deploy your application. Only servers that are registered with the IDE are listed → Click Next.
- Select Struts in the Frameworks panel →Struts option displays in Frameworks panel of New Web Application wizard

CREATING STRUTS2 IN NETBEANS IDE

- The wizard displays the following configuration options:
- **Action Servlet Name**: The name of the Struts action servlet used in the application. The web.xml deployment descriptor contains an entry for the action servlet and specifies the appropriate Struts-specific parameters, such as the path to the servlet class within the Struts library and to the struts-config.xml configuration file within the application.
- **Action URL Pattern**: Specifies the patterns of incoming requests which are mapped to the Struts action controller. This generates a mapping entry in the deployment descriptor. By default, only the *.do pattern is mapped.
- **Application Resource**: Lets you specify the resource bundle which will be used in the struts-config.xml file for localizing messages. By default, this is com.myapp.struts.ApplicationResource.
- **Add Struts TLDs**: Lets you generate tag library descriptors for the Struts tag libraries. A tag library descriptor is an XML document which contains additional information about the entire tag library as well as each individual tag. In general this is not necessary, because you can refer to on-line URIs rather than local TLD files.

CREATING STRUTS2 IN NETBEANS IDE

- Click Finish. The IDE creates the project folder in your file system. As with any web application in the IDE, the project folder contains all of your sources and the IDE’s project metadata, such as the Ant build script.
- The project opens in the IDE. The Projects window is the main entry point to your project sources. It shows a logical view of important project contents.

STRUTS 2 ARCHITECTURE –REQUEST CYCLE

- User sends a request to the server for requesting for some resource (i.e. pages).
- The Filter Dispatcher looks at the request and then determines the appropriate Action.
- Configured interceptor functionalities applies such as validation, file upload etc.
- Selected action is performed based on the requested operation.
- Again, configured interceptors are applied to do any post-processing if required.
- Finally, the result is prepared by the view and returns the result to the user.

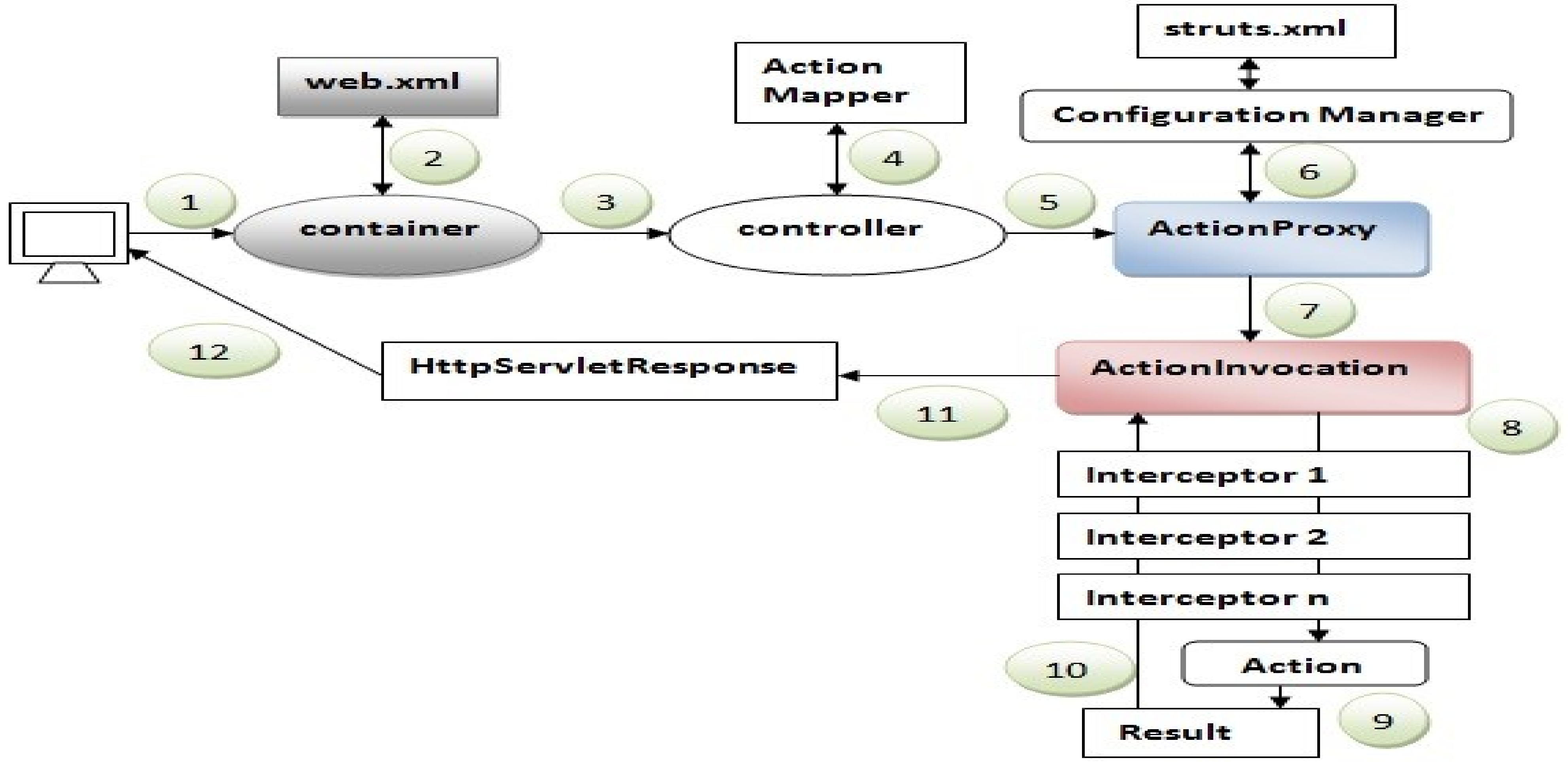
STRUTS 2 ARCHITECTURE

- The **architecture and flow of struts 2 application**, is combined with many components such as Controller, ActionProxy, ActionMapper, Configuration Manager, ActionInvocation, Inerceptor, Action, Result etc.
- To understand the struts flow , there are 2 ways:

1. struts 2 basic flow
2. struts 2 standard architecture and flow provided by Apache struts

UTTARANCHAL UNIVERSITY DEHRADUN NAAC GRADE A+

STRUTS 2 ARCHITECTURE AND BASIC FLOW



UTTARANCHAL UNIVERSITY DEHRADUN NAAC GRADE A+

STRUTS 2 STANDARD FLOW

- 1. User sends a request for the action
- 2. Container maps the request in the web.xml file and gets the class name of controller.
- 3. Container invokes the controller (StrutsPrepareAndExecuteFilter or FilterDispatcher). Since struts2.1, it is StrutsPrepareAndExecuteFilter. Before 2.1 it was FilterDispatcher.
- 4. Controller gets the information for the action from the ActionMapper
- 5. Controller invokes the ActionProxy

STRUTS 2 STANDARD FLOW

6. ActionProxy gets the information of action and interceptor stack from the configuration manager which gets the information from the struts.xml file.
7. ActionProxy forwards the request to the ActionInvocation
8. ActionInvocation invokes each interceptors and action
9. A result is generated
10. The result is sent back to the ActionInvocation
11. HttpServletResponse is generated
12. Response is sent to the user

STRUTS 2 - CONFIGURATION FILES

The web.xml File

- The web.xml configuration file is a J2EE configuration file that determines how elements of the HTTP request are processed by the servlet container.
- It is not strictly a Struts2 configuration file, but it is a file that needs to be configured for Struts2 to work.
- This file provides an entry point for any web application. The entry point of Struts2 application will be a filter defined in deployment descriptor (web.xml).
- Hence we will define an entry of FilterDispatcher class in web.xml. The web.xml file needs to be created under the folder WebContent/WEB-INF.

STRUTS 2 - CONFIGURATION FILES

```
<?xml version = "1.0" Encoding = "UTF-8"?>

<web-app xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"   xmlns =
"http://java.sun.com/xml/ns/javaee"     xmlns:web = "http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"

   xsi:schemaLocation = "http://java.sun.com/xml/ns/javaee     http://java.sun.com/xml/ns/javaee/web-
app_3_0.xsd"

   id = "WebApp_ID" version = "3.0">


   <display-name>Struts 2</display-name>

   <welcome-file-list>

      <welcome-file>index.jsp</welcome-file>

   </welcome-file-list>
```

STRUTS 2 - CONFIGURATION FILES

```
<filter>
    <filter-name>struts2</filter-name>
    <filter-class>  org.apache.struts2.dispatcher.FilterDispatcher  </filter-class>
  </filter>
  <filter-mapping>
    <filter-name>struts2</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>
</web-app>
```

STRUTS 2 - CONFIGURATION FILES-STRUTS.XML

```
<?xml version = "1.0" Encoding = "UTF-8"?>
<!DOCTYPE struts PUBLIC
   "-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
   "http://struts.apache.org/dtds/struts-2.0.dtd">
<struts>
   <constant name = "struts.devMode" value = "true" />
   <package name = "helloworld" extends = "struts-default">
```

STRUTS 2 - CONFIGURATION FILES-STRUTS.XML

```
<action name = "hello"
      class = "com.welcome.struts2.HelloWorldAction"
      method = "execute">
      <result name = "success">/HelloWorld.jsp</result>
    </action>
        <-- more actions can be listed here -->
  </package>
  <-- more packages can be listed here -->
</struts>
```

STRUTS 2 - CONFIGURATION FILES-STRUTS.XML

- The package tag has the following attributes:
- **name (required)-** The unique identifier for the package
- **Extends -** which package does this package extend from. By default, we use struts-default as the base package.
- **Abstract -**If marked true, the package is not available for end user consumption.
- **Namespace -**Unique namespace for the actions

THE STRUTS-CONFIG.XML FILE

- The configuration file basically contains following main elements :
- **struts-config -**This is the root node of the configuration file.
- **form-beans -**This is where you map your ActionForm subclass to a name. You use this name as an alias for your ActionForm throughout the rest of the strutsconfig.xml file, and even on your JSP pages.
- **global forwards -**This section maps a page on your webapp to a name. You can use this name to refer to the actual page. This avoids hardcoding URLs on your web pages.

STRUTS 2 - CONFIGURATION FILES-STRUTS.XML

- **action-mappings -**This is where you declare form handlers and they are also known as action mappings.
- **Controller -**This section configures Struts internals and rarely used in practical situations.
- **plug-in-**This section tells Struts where to find your properties files, which contain prompts and error messages

ACTION CLASS

- Action Class in Struts framework defines the business logic. An action class handles the client request and prepares the response. It also decides where the response should be forwarded.

- Basically an action class receives data from the presentation layer and forwards the data to the corresponding business layer. It also processes the data which comes from the business layer and forwards them back to the presentation layer.

- Action classes are used to invoke the classes at the business layer or data access layer to get the data from the bean and store the processed data and return the result or error depending upon the situation.

ACTIONS

- The two other important capacities.
- Firstly, the action plays an important role in the transfer of data from the request through to the view, whether its a JSP or other type of result.
- Secondly, the action must assist the framework in determining which result should render the view that will be returned in the response to the request.vice the request from the user.

CREATE ACTION

- The only requirement for actions in **Struts2** is that there must be one no argument method that returns either a String or Result object and must be a POJO.
- If the no-argument method is not specified, the default behavior is to use the execute() method.
- Optionally you can extend the **ActionSupport** class which implements six interfaces including **Action** interface

CREATE ACTION

```
public interface Action {
   public static final String SUCCESS = "success";
   public static final String NONE = "none";
   public static final String ERROR = "error";
   public static final String INPUT = "input";
   public static final String LOGIN = "login";
   public String execute() throws Exception;
}
```

CREATE ACTION

```
public class HelloWorldAction {
  private String name;
  public String execute() throws Exception {
    return "success";
  }
    public String getName() {
    return name;
  }
  public void setName(String name) {
    this.name = name;
  }   }
```

ACTION SUPPORT

```
import com.opensymphony.xwork2.ActionSupport;
public class HelloWorldAction extends ActionSupport {
  private String name;
  public String execute() throws Exception {
    if ("SECRET".equals(name)) {
      return SUCCESS;
    } else {
      return ERROR;
    }
  }
```

ACTION SUPPORT

```
public String getName() {
    return name;
  }

  public void setName(String name) {
    this.name = name;
  }
}
```

STRUTS.XML FOR HELLOWORLD

```
<struts>
   <constant name = "struts.devMode" value = "true" />
   <package name = "helloworld" extends = "struts-default">
      <action name = "hello"   class = "com.welcome.struts2.HelloWorldAction“
method = "execute">
         <result name = "success">/HelloWorld.jsp</result>
         <result name = "error">/AccessDenied.jsp</result>
      </action>
   </package>
</struts>
```

CREATE A VIEW – HELLOWORLD.JSP

```
<%@ page contentType = "text/html; charset = UTF-8" %>
<%@ taglib prefix = "s" uri = "/struts-tags" %>
<html>
   <head>
      <title>Hello World</title>
   </head>
   <body>
      Hello World, <s:property value = "name"/>
   </body>
</html>
```

ACCESSDENIED.JSP

```
<%@ page contentType = "text/html; charset = UTF-8" %>
<%@ taglib prefix = "s" uri = "/struts-tags" %>
<html>
   <head>
      <title>Access Denied</title>
   </head>
   <body>
      You are not authorized to view this page.
   </body>
</html>
```

INDEX.JSP

```
<%@ page language = "java" contentType = "text/html; charset = ISO-8859-1"
  pageEncoding = "ISO-8859-1"%>
<%@ taglib prefix = "s" uri = "/struts-tags"%>
  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>Hello World</title>
  </head>
    <body>
```

INDEX.JSP

```
<h1>Hello World From Struts2</h1>
    <form action = "hello">
      <label for = "name">Please enter your name</label><br/>
      <input type = "text" name = "name"/>
      <input type = "submit" value = "Say Hello"/>
    </form>
  </body>
</html>
```

INTERCEPTORS

- Interceptors are conceptually the same as servlet filters or the JDKs Proxy class.
- Interceptors allow for crosscutting functionality to be implemented separately from the action as well as the framework.
- Interceptors helps in –
- Providing preprocessing logic before the action is called.
- Providing postprocessing logic after the action is called.
- Catching exceptions so that alternate processing can be performed.

RESULT TYPE

- The **<results>** tag plays the role of a **view** in the Struts2 MVC framework. The action is responsible for executing the business logic.
- The next step after executing the business logic is to display the view using the **<results>** tag.
- Struts comes with a number of predefined **result types** and the default result type is **dispatcher**, which is used to dispatch to JSP pages.
- Struts allow you to use other markup languages for the view technology to present the results and popular choices include **Velocity, Freemaker, XSLT** and **Tiles**.

VALIDATIONS

- At the Struts core, we have the validation framework that assists the application to run the rules to perform validation before the action method is executed.
- Client side validation is usually achieved using Javascript. However, one should not rely upon client side validation alone.
- The best practices suggest that the validation should be introduced at all levels of your application framework.
- When the user presses the submit button, Struts 2 will automatically execute the validate method and if any of the "if" statements listed inside the method are true, Struts 2 will call its **addFieldError** method.
- If any errors have been added, then Struts 2 will not proceed to call the execute method. Rather the Struts 2 framework will return input as the result of calling the action.

VALIDATIONS

- Since, we used Struts 2 form tags, Struts 2 will automatically add the error messages just above the form filed.
- These error messages are the ones we specified in the addFieldError method call.
- The addFieldError method takes two arguments. The first, is the **form** field name to which the error applies and the second, is the error message to display above that form field.